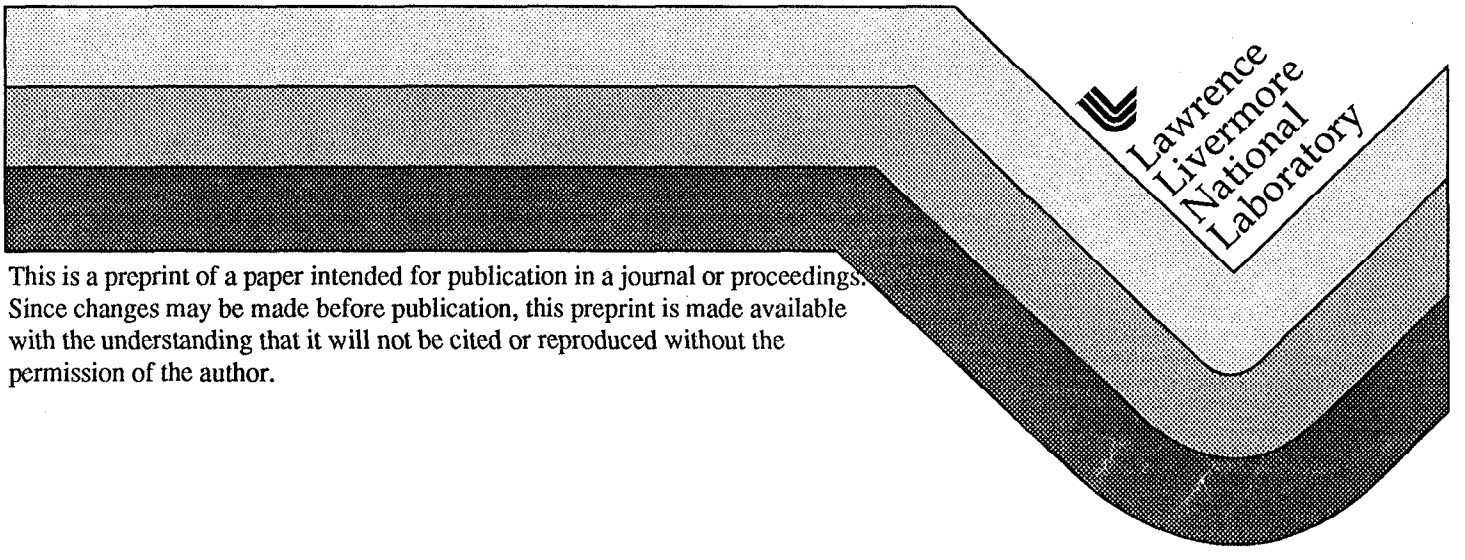
UCRL-JC-131746
Preprint

Coupled Mechanical/Heat Transfer Simulation on MPP Platforms Using a Finite Element/Linear Solver Interface

Collin J. Aro
Evi I. Dube
W. S. Futral

This paper was prepared for submittal to
Ninth Society for Industrial and Applied Mathematics
Conference on Parallel Processing for Scientific Computing
San Antonio, TX
March 22-24, 1999

February 24, 1999



Coupled Mechanical/Heat Transfer Simulation on MPP Platforms Using a Finite Element/Linear Solver Interface*

Colin J. Aro, Evi I. Dube, W. Scott Futral†

Abstract

This report describes the implementation of a coupled mechanical/heat transfer simulation using a Finite Element Interface (FEI). The FEI is an abstraction layer, which lies between the application code and its linear solver libraries, controlling the set-up and solution of the linear system arising in the finite element simulation. The performance and scalability of the ISIS++ FEI is examined on the ASCI Red and Blue machines in the context of the ALE3D finite element simulation code.

1.0 Introduction

Complex, highly resolved, 3-D multi-physics simulations, such as those being developed for the Department of Energy's Accelerated Strategic Computing Initiative (ASCI), require the use of state-of-the-art massively parallel (MPP) computers. The sheer size and computational complexity of these applications demand the efficient implementation of scalable multi-physics and linear solver algorithms in order to maximize the performance of the MPP architectures on which they run. At the same time, concerns such as ease of development and portability demand standardized interfaces (libraries) for the management of certain resources, with MPI[1] being a prime example of such an interface.

Linear solvers are key to many of these simulations, often forming the computational kernel of the application. As such, the data structures and implementation details of a particular solver package tend to permeate the application to a very large degree. For example, the matrix decomposition across a distributed memory architecture is influenced by the matrix data storage format. At the same time, the matrix decomposition is often derived from the physical domain decomposition used in the application code. Furthermore, when slide surfaces (contact surfaces) are present, the optimal domain decomposition for the slide surfaces may differ significantly from that of the physical problem, and may further differ from the preferred layout for the matrix solver. The code developer must deal with these issues in forming the global matrix and distributing it across processors, keeping in mind the particulars of the linear solver package being used.

Having the details of a particular linear solver permeated throughout an application code in this way leads to portability problems and makes the task of changing or modifying the linear solver algorithm difficult and time consuming. The ISIS++ Finite Element Interface (FEI) being developed at Sandia National Laboratory is designed to address this issue[2][3]. The FEI provides an abstraction layer between the finite element client appli-

*. This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-Eng-48.

†. Lawrence Livermore National Laboratory, P.O. Box 808, L-098, Livermore, CA, 94551

cation and the linear solver library. The application code must provide the FEI with mesh geometry, connectivity information, domain decomposition, elemental stiffness matrices, physical boundary conditions, and contact information. The FEI then allocates, distributes and assembles the global matrix. The application can choose from a selection of linear solver packages implemented under the FEI. The solution values are returned to the application in a node-based‡ fashion. This abstraction (or translation) layer provides complex 3-D simulation codes portable plug-and-play ability with a variety of linear solvers in a parallel computing environment, facilitating rapid linear solver research.

In this paper, a heat transfer simulation module is developed for the ALE3D code using the ISIS++ FEI. The performance and scalability of the FEI will be examined. Depending on the problem, the parallel performance of the simulation code can be strongly influenced by the parallel performance of the FEI. Therefore, understanding the FEI's scalability is important in order to tune implicit applications for peak performance. Results will be presented of the ASCI Red and Blue machines.

1.1 Overview of ALE3D

ALE3D[4][5] is a finite element code being developed at the Lawrence Livermore National Laboratory that treats fluid and elastic-plastic response on an unstructured grid. The grid may consist of arbitrarily connected hexahedra (Figure 1), and the mesh can be constructed from disjoint blocks of elements which interact at the boundaries via slide surfaces and other types of boundary conditions. Nodes can be designated as relax nodes and ALE3D will adjust their position relative to the material in order to relieve distortion or to improve accuracy or efficiency. This relaxation process can allow nodes to cross material boundaries and create mixed or multi-material elements

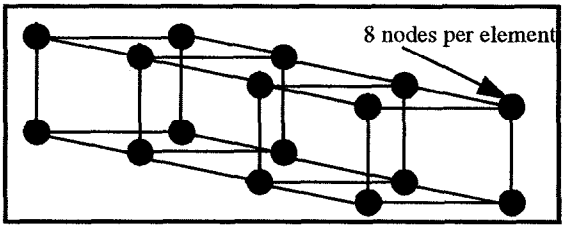


FIGURE 1. The unstructured grid used by ALE3D consists of eight-node bricks.

The basic computational step consists of a Lagrangian step followed by an advection (or remap) step. In the Lagrangian phase, nodal forces are accumulated and an updated nodal acceleration is computed. The stress gradients and strain rates are evaluated by a lowest order finite element method. A diagonal mass matrix is used. Second order accuracy is obtained with a grid that is staggered in both space and time.

The advection (remap) step allows for either a pure Eulerian calculation, in which the nodes are placed back in their original positions, or a more complex scheme involving mesh relaxation techniques. This nodal motion or relaxation generates inter-element fluxes which must be used to update velocities, energies, masses, stresses and other constitutive properties. This remapping process is referred to as advection. Second order accurate schemes are required to perform this operation with sufficient accuracy. Additionally,

‡. To avoid confusion, "nodes" will always refer to the finite element mesh. When speaking of processing nodes in a parallel environment, "processing nodes" will always be used.

it is generally inadequate to allow advection only within material boundaries. ALE3D has the ability to treat multi-material elements, thus allowing relaxation to take place across material boundaries.

The interaction at slide surfaces can consist of pure sliding in which there are no tangential forces on interface nodes. Alternatively, the nodes may be tied to inhibit sliding entirely, a coulomb friction algorithm may be used, or turbulence models may be applied. Voids may open or close between the surfaces depending on the dynamics of the problem, and there is an option to allow a block to fold back on itself (single-sided sliding). Where no void is present, the forces on either side of the slide surface are accumulated and used to produce a net acceleration of the nodes on the surface consistent with the center-of-mass motion. In this manner, the dynamics of both the fluid and structure are treated in a consistent manner. The ability to remove slide surfaces allows for the flexibility for advecting across these boundaries.

ALE3D is an example of the next generation of ASCI codes, simulating safety and manufacturing problems. In order for the code to model these types of problems, new code physics, including heat transfer modeling capability, must be added to accurately predict the responses to hazard scenarios and manufacturing needs.

1.2 Heat Transfer Formulation

The heat transfer formulation begins with the heat diffusion equation:

$$\rho c_v \frac{\partial T}{\partial t} = \nabla\bullet(\kappa\nabla T) + \dot{q}$$

In this equation, the scalars ρ, c_v, and T represent density, specific heat, and temperature. The term $\dot{q}$ is a scalar heat source, while κ is a symmetric thermal conductivity tensor. The domain of definition is Ω, a bounded subdomain of R^3, with appropriate conditions prescribed on the boundary, Γ.

The heat diffusion equation is multiplied by a test function, ϕ_i, and integrated over Ω:

$$\int_\Omega \left(\rho c_v \frac{\partial T}{\partial t}\right)\phi_i d\Omega - \int_\Omega [\nabla\bullet(\kappa\nabla T)]\phi_i d\Omega - \int_\Omega \dot{q}\phi_i d\Omega = 0$$

The divergence theorem is applied after integrating the conduction term by parts, so that

$$\int_\Omega \left(\rho c_v \frac{\partial T}{\partial t}\right)\phi_i d\Omega + \int_\Omega (\kappa\nabla T \bullet \nabla\phi_i) d\Omega = \oint_\Gamma (\phi_i \kappa\nabla T)\bullet \hat{n} d\Gamma + \int_\Omega \dot{q}\phi_i d\Omega$$

Finally, $\gamma = \kappa\nabla T \bullet \hat{n}$ is defined to be the heat flux through the boundary, Γ. This results in

$$\int_\Omega \left(\rho c_v \frac{\partial T}{\partial t}\right)\phi_i d\Omega + \int_\Omega (\kappa\nabla T \bullet \nabla\phi_i) d\Omega = \oint_\Gamma \gamma\phi_i d\Gamma + \int_\Omega \dot{q}\phi_i d\Omega$$

and is the integral form of the equation used by the ALE3D finite element formulation.

The Galerkin formulation is used. The temperature is written as a linear sum of basis functions:

$$T(x, y, z, t) = \sum_{j=1}^{nodes} \phi_j(x, y, z)\hat{T}_j(t)$$

Substitution into the integral form of the heat transfer equation gives

$$\sum_{j=1}^{nodes} \left(\int_{\Omega} \rho c_v \phi_i \phi_j d\Omega \right) \frac{d\hat{T}_j}{dt} + \sum_{j=1}^{nodes} \left(\int_{\Omega} \nabla\phi_i \bullet \kappa \nabla\phi_j d\Omega \right) \hat{T}_j = \int_{\Omega} \dot{q}\phi_i d\Omega + \oint_{\Gamma} \phi_i \gamma d\Gamma$$

which defines a set of linear equations. This can be written in matrix form by defining

$$M_{ij} = \int_{\Omega} \rho c_v \phi_i \phi_j d\Omega$$

$$N_{ij} = \int_{\Omega} \nabla\phi_i \bullet \kappa \nabla\phi_j d\Omega$$

$$f_i = \int_{\Omega} \dot{q}\phi_i d\Omega + \oint_{\Gamma} \phi_i \gamma d\Gamma$$

so that

$$M\frac{d\hat{T}}{dt} + N\hat{T} = f$$

To model transient behavior, $\hat{T}$ is discretized in time and integrated:

$$M\Delta\hat{T}^n + \Delta t N(\hat{T}^n + \alpha\Delta\hat{T}^n) = \Delta t f$$

or

$$(M + \alpha\Delta t N)\Delta\hat{T}^n = \Delta t(f - N\hat{T}^n) \tag{1}$$

This defines the linear system to be solved at each timestep. Note that it includes the explicit Euler ($\alpha = 0$), implicit Euler ($\alpha = 1$), and Crank-Nicholson ($\alpha = 0.5$) schemes. It is further possible to deal with nonlinearities by defining an outer Newton-Raphson iteration in the event c_v, κ, or $\dot{q}$ are functions of temperature. The computations in this study use the trilinear basis functions

$$\phi_i(\xi, \eta, \zeta) = \frac{1}{8}(1 + \xi_i\xi)(1 + \eta_i\eta)(1 + \zeta_i\zeta)$$

The integrals needed to fill the matrices are computed using second order Gaussian quadrature.

1.3 Mechanical/Heat Transfer Coupling

The ALE3D code currently uses an operator split approach to multi-physics simulation, applying a series of alternating mechanical and thermal steps. The mechanical steps move the nodes while holding the entropy fixed. The heat transfer steps move heat between the

nodes while holding nodal positions fixed. The mechanical energy is modified by the change induced by the heat transfer step. Likewise, the thermal energy must be modified by the mechanical step, which conducts a change in volume and strain.

For the constant entropy change in volume, consider the change in temperature from thermodynamics:

$$\left(\frac{\partial T}{\partial V}\right)_S = -\left(\frac{\partial P}{\partial S}\right)_V = -T\left(\frac{\partial P}{\partial E}\right)_V = -\frac{T\gamma}{V}$$

where γ is the Gruniesen gamma function. Similarly, the elastic stress-strain component is obtained by changing the material deviatoric strain $\hat{\varepsilon}$ while holding the entropy fixed:

$$\left(\frac{\partial T}{\partial \hat{\varepsilon}}\right)_S = TV\left(\frac{\partial \hat{\xi}}{\partial E}\right)_{\hat{\varepsilon}} = 2TV\left(\frac{\partial \mu}{\partial E}\right)_{\hat{\varepsilon}} \hat{\varepsilon}$$

Here, $\hat{\xi}$ is the deviatoric stress, while μ is the shear modulus. The volume and strain contributions are combined into a single parameter ψ which is passed from the mechanical step to the thermal step:

$$\frac{\Delta T}{T} = \psi = -\gamma\frac{\Delta V}{V} + 2V\left(\frac{\partial \mu}{\partial E}\right)_{v\hat{\varepsilon}} (\hat{\varepsilon} \cdot \Delta\hat{\varepsilon})$$

Another mechanism used to influence the temperature change is to add energy directly to the thermal equations, via the source terms. This mechanism is used only for plastic work, where all of the plastic work energy is deposited as thermal energy. The advantages of the parametrized method over the direct addition of additional energy is that it is guaranteed to always result in a positive temperature[6], and that the data passed between modules is unit-less, thereby reducing the complexity required.

2.0 Implementation

ALE3D uses a domain decomposition paradigm so that each processor in the MPP environment owns a subcollection of elements. While elements are owned by distinct processors, individual nodes must often be shared (Figure 2). As a practical matter, the lists of elements and nodes living on a processor are readily available, as is the list of shared nodes and neighbor processors.

For the setup and solution of the linear system (Eq. 1), the FEI consists of four main steps: initialization, loading, solution, and solution return. The first step in the process, initialization, teaches the FEI about the structure of the finite element data, so that it may translate this physical structure into an algebraic structure for the underlying sparse matrix. The data passed to the FEI in this phase includes control data defining the types of elements used (number of nodes, degrees of freedom), raw element data (element and node IDs, connectivity information), control data for special nodes (boundary conditions, shared nodes), and constraint relation data (slide surfaces).

Once the sparse matrix structure has been determined, it can then be populated with finite element data according to the standard finite element matrix assembly process. This is the

task of the load step. The data passed to the FEI during this stage includes element arrays (stiffness matrices and load vectors), boundary condition data, and constraint relation data. At the end of this step, the global matrix is fully assembled and has been modified to account for all appropriate boundary conditions and constraint relations.

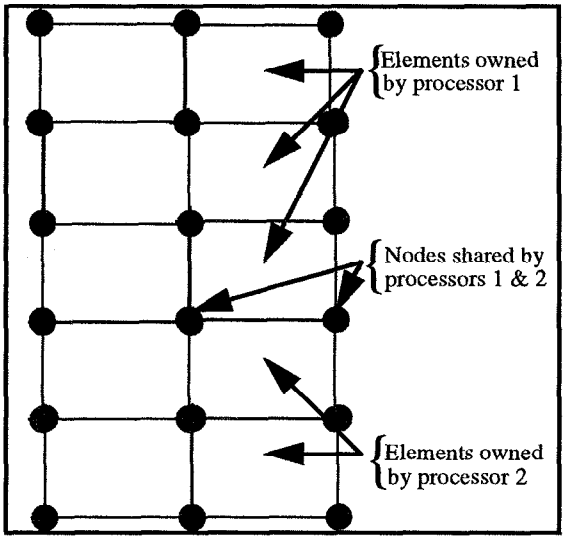


FIGURE 2. The ALE3D domain (grid) decomposition assigns elements to unique processors, but nodes along domain boundaries must be shared. Here the red and green elements belong to separate processors, while the blue nodes are shared.

The solution step requires control data such as convergence tolerance, maximum iteration count, and solver/preconditioner pair. Some of this data is necessarily solver dependent, although the FEI seeks to provide reasonable defaults whenever possible. After the solver is invoked, the solution return step converts the raw algebraic data into corresponding finite element data, which is returned in a node-based fashion.

The data needed by the FEI in each of the four stages is collected in parallel and can be passed as aggregate blocks of elements to permit efficient use of cache memory. The matrix is distributed across processors using a row matrix abstraction with the decomposition based on the original problem domain decomposition. In addition to its natural application-oriented interface, which allows easy and bug-free setup of finite element problems in an MPP environment, the FEI's runtime solver/preconditioner selection permits rapid linear solver research in areas where improvement in current solver technology is needed.

3.0 Computational Results

The parallel performance and scalability of the FEI will be examined in this section. The test problem will be a 3-D block of aluminum undergoing volumetric heating. The block's position will be fixed at a single node so that the heating causes expansion in the material away from the fixed node. The mechanical expansion will result in adiabatic cooling, so that both the mechanical-thermal and thermal-mechanical couplings will be exercised. Although it contains a relatively simple geometry, this problem is well suited for the task at hand, since the scalability of the FEI's tasks (save the itertative solution of the matrix itself) is unaffected by the geometry of the problem. For the matrix solution, a diagonally scaled conjugate gradient iteration is extremely effective, often converging in four iterations for a million zone version of this problem. Thus, for a problem such as this one (i.e. for a problem with an "easy" matrix), the total wall clock time will be dominated by the FEI time, which makes it well suited for this study. For problems with more challenging matrices, the FEI accounts for less of the overall wall clock time, but not such a small fraction that its performance can be totally ignored.

This problem is linear, so no Newton-Raphson outer iteration is used. The implicit Euler method is used for thermal timestepping, while the explicit mechanical package is used with a density scaling option[6] to permit large timesteps without a full implicit mechanical calculation. Two grid resolutions will be employed for the block, which measures one cubic meter: 125 thousand elements (the small block) and one million elements (the big block).

The architectures used are the ASCI Red and Blue machines. ASCI Red[7] is a network of over four thousand dual-processor 200 Mhz Pentium Pro chips, with 256K of cache memory each. The results presented here use only a single processor per processing node, so that the effective processor-to-processor bi-directional bandwidth is 800 megabytes per second. ASCI Blue-Pacific[8] consists of 1464 4-way SMP processing nodes divided into three "sectors." The processing nodes can accommodate up to four MPI processes each, but these processes are then restricted to the slower IP message passing protocol. This leads to additional contention between same processing node MPI processes and can result in poor scalability. The processors are 332 Mhz 604e PowerPC chips with 1.5 gigabytes of memory per processing node (i.e. shared by four processors). The bandwidth between processing nodes is 150 megabytes per second. However, when the slower IP mode is used, the point-to-point bandwidth drops to 25 megabytes per second, with aggregate performance (all four processes) at 47 megabytes per second (i.e. all four processes cannot achieve 25 Mb/sec). This penalty in communication performance will be evident when the FEI scaling is examined.

3.1 Scaling for Fixed Problem Size

In this section, the parallel scaling of the FEI will be examined on the small block. This is carried out by decomposing the fixed size block into increasingly more subdomains and running with an increasing number of processors. The ALE3D code is run with a single domain per processor for the results presented here.

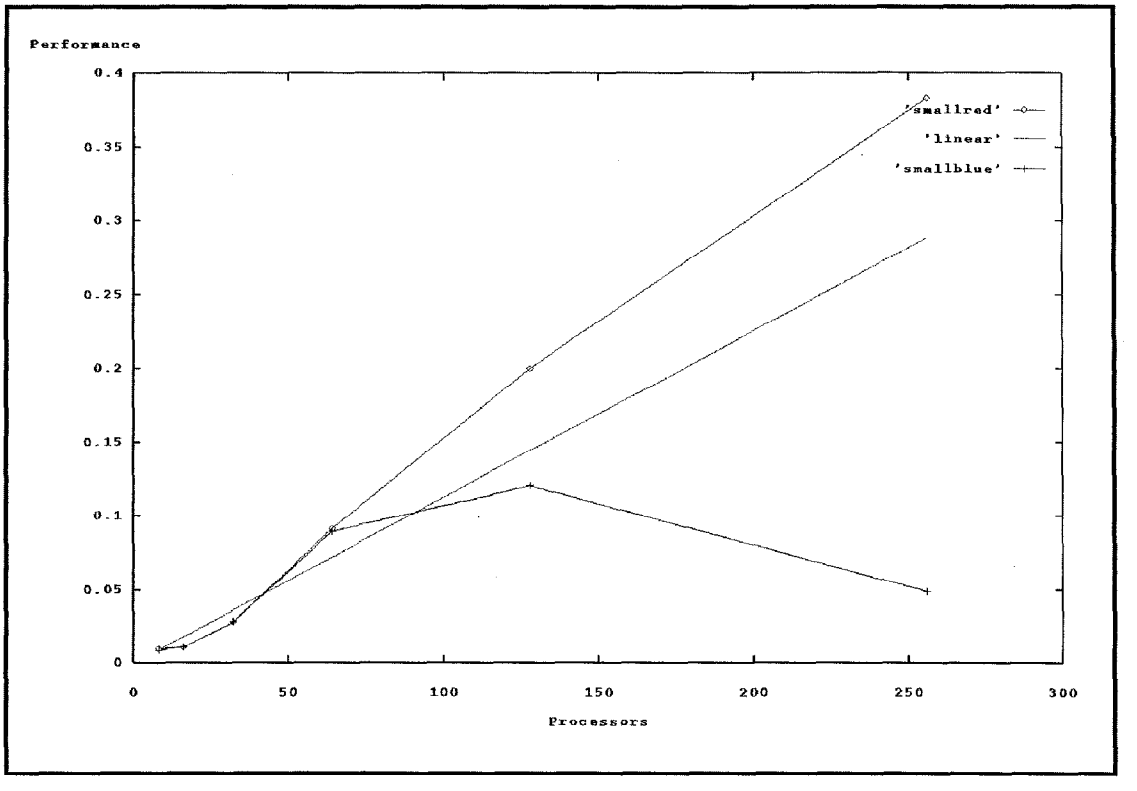
Table 1 lists average CPU time spent in FEI routines (excluding the matrix solve). This data is gathered as an average over 10 cycles. The statistical standard deviation is given in column 2, while column 3 shows the "instantaneous" speedup over the preceding run (with half as many processors).

TABLE 1. Average FEI times for the small block running on the ASCI architectures

	Red Machine			Blue Machine		
Procs.	**FEI time**	**Std. Dev.**	**Speedup**	**FEI time**	**Std. Dev.**	**Speedup**
8	104.99	0.15	N/A	113.27	0.35	N/A
16	95.84	0.19	1.10	95.08	0.71	1.19
32	35.47	0.02	2.70	35.44	0.42	2.68
64	10.92	0.01	3.25	11.17	0.25	3.17
128	5.00	0.01	2.18	8.30	0.89	1.35
256	2.61	0.01	1.92	20.45	2.12	0.41

On the ASCI Red machine, the small block does not scale well using a small number of processors, but scales superlinearly with 32 or more processors. This trend continues all the way to 256 processors. The FEI is somewhat communication bound, so the Red machine's large communication bandwidth is paying obvious dividends here.The ASCI Blue architecture behaves identically, until reaching 128 processors. The scaling falls off rather dramatically thereafter, an obvious consequence of the low communication bandwidth when running in IP mode. When run with a single processor per processing node (user space, or US mode), the numbers produced by Blue have similar characteristics to those on Red. The obvious drawback to the US mode, however, is the waste of computing resources and the resulting smaller processor partition that results. In Figure 3, the performance (inverse of FEI time) is plotted along with a linear performance curve (for reference).

FIGURE 3. Performance plots for the ASCI Red and Blue machines on the small block.



This result indicates which processor loadings are likely to be most efficient for larger problems. On the Blue architecture, three to five thousand elements per processor appears to deliver good FEI efficiency, while on the Red machine, the FEI performs efficiently with as few as 500 elements per processor.

3.2 Scaling for Fixed Processor Load

In this section, the previous result will be repeated on the big block. This will provide information as to how the FEI scales with a fixed number of finite elements per processor, running increasingly bigger (finer resolution) simulations. The big block is identical to the

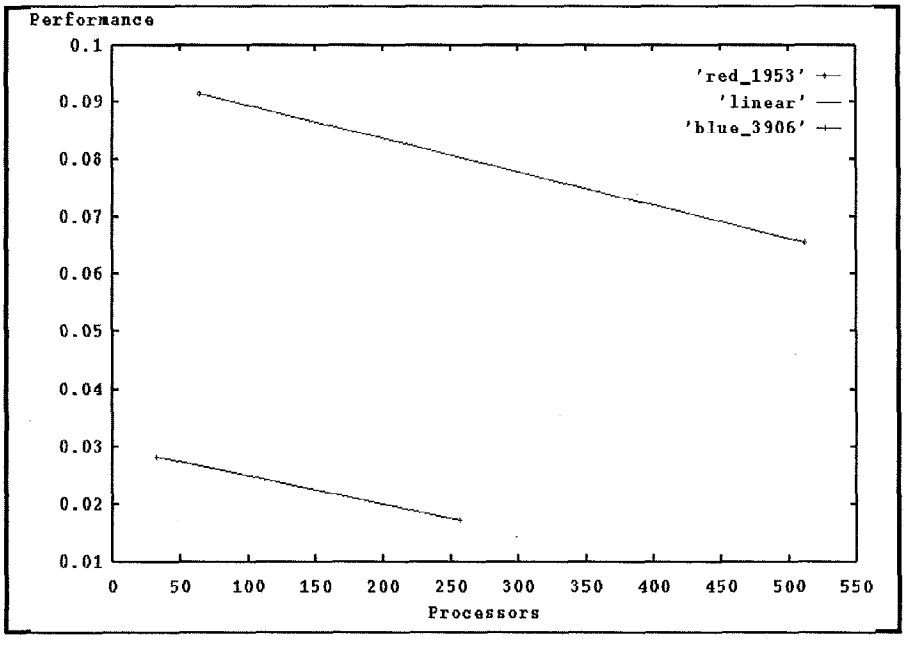
small block save the finer resolution. It is used to scale the problem size up and repeat the previous result. Runs analogous to those in Table 1 are presented in Table 2.

TABLE 2. Average FEI times for the large block running on the ASCI architectures

	Red Machine			Blue Machine		
Procs.	**FEI time**	**Std. Dev.**	**Speedup**	**FEI time**	**Std. Dev.**	**Speedup**
64	167.054	0.04	N/A	313.00	47.39	4.45
128				139.81	10.97	2.24
256				58.16	1.53	2.40
512	15.260	0.00	10.95	75.88	6.86	0.77

As with the small block, the problem scales well when the number of processors is varied. The ASCI Blue architecture, again running 4 processors per processing node, stops scaling at approximately the same processor loading: three to five thousand elements per processor. In terms of a constant processor load with increasing problem size, however, the problem does not scale well. The best performance achieved by the Blue machine occurs in the 32 domain small run and the 256 domain large run. This results in approximately 3900 finite elements per processor. The FEI time increases by 65% for the larger calculation (Figure 4).

FIGURE 4. Constant processor load scaling on ASCI Red and Blue. The Red run uses approximately two thousand elements per processor, while the Blue run uses four thousand elements per processor.

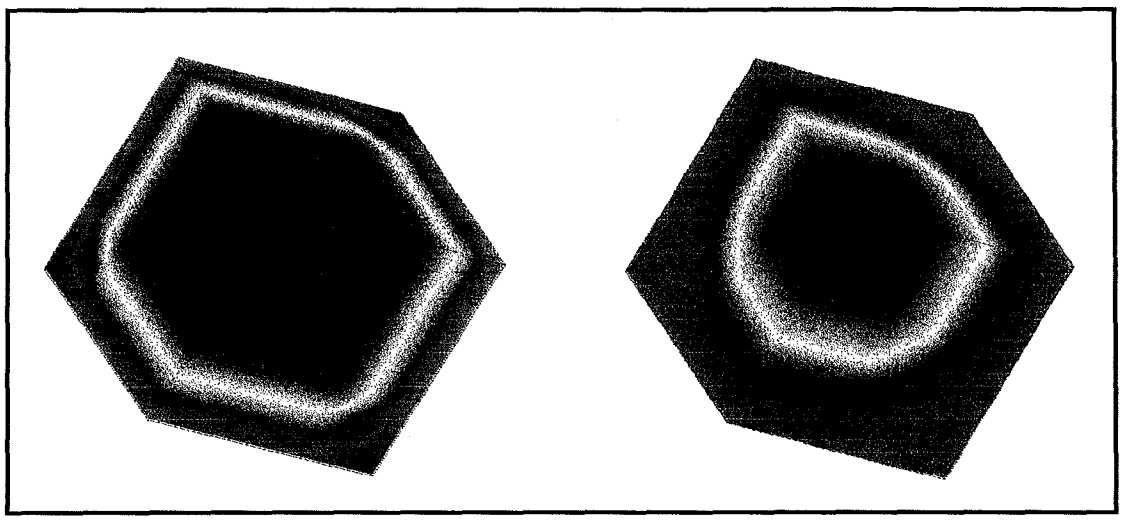


For the Red machine, data points were especially hard to come by due to the small memory capacity and low I/O performance. Nonetheless, constant load scaling was better than the Blue machine. At 15,625 elements per processor, FEI time increases by 60% (better than the Blue machine's best case) for the larger calculation. At approximately two thou-

sand elements per processor, the larger simulation requires 40% more FEI time. Note that this behavior cannot be due to the matrix solve, which is very simple; it is the interface that is not scaling well in this situation. Figure 4 plots the performance (inverse of FEI time) achieved by the Blue machine at approximately 3900 elements per processor and the Red machine at approximately two thousand elements per processor.

To investigate an extended simulation, the Blue machine (256 processors) was run for 500 cycles. On average, the FEI time was 57.91 seconds with a standard deviation of 3.70 seconds. This result is exactly in line with the FEI times quoted in Table 2 over an extended simulation. The output from this simulation is presented in Figure 5, showing the heating, expansion, and subsequent cooling of the block. This illustrates a high-resolution, multi-physics, application implemented on an MPP architecture. It's rapid development and portability demonstrate the dividends paid by the FEI concept.

FIGURE 5. Temperature plot for the million-element calculation on ASCI Blue. The outer faces are expanding and cooling (blue areas) while the fixed node is the last to cool (red area).



4.0 Conclusions

This study has examined the performance of the ISIS++ FEI running million element simulations on ASCI MPP architectures. It has shown the FEI to be communication limited, with poor scaling results when inter-processor communication bandwidth is low. Regardless, the FEI still scales well when each processor is given three to five thousand finite elements. This result holds for both the ASCI Red and Blue machines. For applications with relatively simple matrices, this result is important in order to achieve good efficiency. For applications with more challenging matrices, the FEI time is a smaller percentage of the overall wall clock time, but cannot be ignored when fine tuning implicit applications for peak performance on MPP platforms. Finally, the FEI concept has proven to be useful, cutting the time and the effort required to develop, port, or modify the linear solver portions of the code.

Research will continue to investigate FEI performance on more difficult simulation geometries, in particular those with slide surfaces and those with implicit mechanical modeling

requirements. For these types of problems, fine resolution, sliding surfaces, distorted elements, and regions of low material strength produce a very challenging matrix, so the solvers must be as finely tuned as the FEI, in order to permit an efficient simulation.

5.0 Acknowledgments

The authors would like to thank the ISIS++ development team: Robert Clay and Alan Williams of Sandia National Lab and Kim Mish of Chico State University. We are also grateful for assistance from Juliana Hsu, Rose McCallen, Albert Nichols, and the rest of the ALE3D team at LLNL.

6.0 References

[1] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: Portable Parallel Programming with the Message-Passing Interface*, MIT Press, Cambridge, MA, 1994

[2] R.L. Clay, K.D. Mish, and A.B. Williams, *ISIS++ Reference Guide (Iterative Scalable Implicit Solver in C++) Version 1.0*, Sandia National Laboratory, Livermore, CA, USA, 1997 (unpublished)

[3] R.L. Clay, K.D. Mish, I.J. Otero, L.M. Taylor, and A.B. Williams, *An Annotated Reference Guide to the Finite Element Interface Specification for Use with Scalable Equation Solver Packages Version 1.0*, Sandia National Laboratory, Livermore, CA, USA, 1998 (unpublished)

[4] R. Sharp, S. Anderson, E. Dube, and I. Otero, *Users Manual for ALE3D*, Lawrence Livermore National Laboratory, Livermore, CA, USA, 1993 (unpublished)

[5] R. Couch, R. Sharp, I. Otero, R. Tipton, and R. McCallen, *Application of ALE Techniques to Metal Forming Simulations*, in AMD-Vol. 180 Advanced Computational Methods for Material Modeling, D.J. Benson and R.J. Asaro eds., The American Society of Mechanical Engineers, New York, NY, USA 1993, pp. 133-140

[6] A.L. Nichols III, R. Couch, R.C. McCallen, I. Otero, and R. Sharp, *Modelling Thermally Driven Energetic Response of High Explosive*, Proceedings of 11th International Detonation Symposium, Snowmass, CO, Office of Naval Research, 1998

[7] T.G. Mattson, D. Scott, S. Wheat, *The ASCI Red, TFLOPS Supercomputer*, Intel Corp., Beaverton, OR, USA, 1996 (http://mephisto.ca.sandia.gov/TFLOP/sc96/index.html) (unpublished)

[8] ASCI Blue-Pacific: *System Attributes for the ASCI Blue-Pacific Systems*, Livermore, CA, USA, 1997 (http://www.llnl.gov/asci/platforms/bluepac/blue.table.html) (unpublished)